

System upgrade with SWUpdate

ELC 2017

02/2017

Gabriel Huau

Embedded Software Engineer

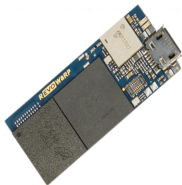


TABLE OF CONTENTS

1. Introduction
2. Architecture
3. Customization
4. Integration

ABOUT THE PRESENTER

- Open-Source Enthusiast
 - Buildroot / Linux kernel / U-Boot / ...
- Working @ [Witekio](#)
- Android, Linux and QNX:
 - BSP adaptation
 - Driver development
 - System integration





Introduction

WHY ARE WE HERE?

- Importance of product updates in the field
 - Bug / Security fixes
 - New features
- What's different in embedded systems?
 - Power-safe
 - Access to target



WHY ARE WE HERE?

- Focusing on [SWUpdate](#)
 - Update **framework**
 - Open-Source (of course)
 - Created & maintained by [Stefano Babic](#) from Denx
- Cover all aspects
 - From update creation to download to flashing
- Past experiences
 - Customers / Users feedback
- Practical approach
 - Demonstration on actual HW



Architecture

WHAT TO UPDATE?

- Bootloader
 - ▶ Not covered in this talk
 - ▶ Depends on HW capabilities
- Kernel
- Device tree
- Root file system
- Application data

ATOMIC UPDATE VS PACKAGE MANAGER

Package manager pros:

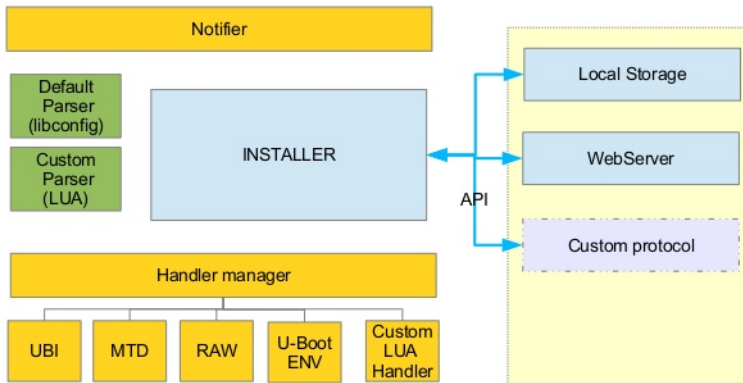
- small update image

Package manager cons:

- upgrade not atomic (in general ...)
- hard for testing and support
- more places where things can go wrong

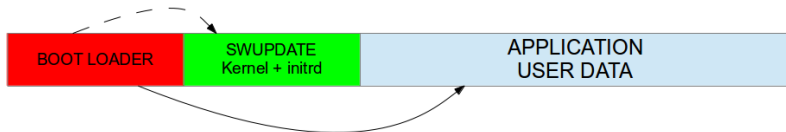
SWUPDATE OVERVIEW

Swupdate architecture



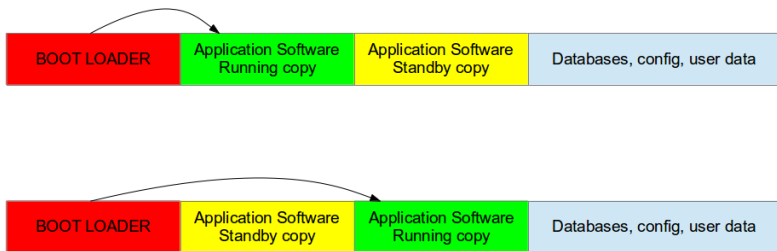
PARTITIONS LAYOUT

- **Single copy** - running as standalone image
 - ▶ Consists of kernel / dt + initrd
 - ▶ Much smaller than entire system
 - ▶ Bootloader in charge of loading standalone image
 - ▶ System must reboot to enter update process



PARTITIONS LAYOUT

- **Double copy** with fall-back
 - Requires twice as much space
 - Guarantees there's always a working copy!
 - Bootloader in charge of booting proper image



SWUPDATE TOOL

- Runs / applies update from **Linux user-space**
 - More generic than bootloader
 - More drivers / protocols supported
 - Lots of tools / libraries available
- **Full update** only
 - Atomic process
 - Single image delivery
- Small footprint: compressed ramdisk of 4MB (could be used for rescue image)

AVAILABLE FEATURES

- Update interfaces
 - Local
 - ◆ USB, SD, UART, etc...
 - OTA / Remote
 - ◆ HTTP / web based / HawkBit
- **Security** (hash, signature, etc...)
- Standard parser with many handlers
 - Images to be installed; can be compressed
 - Scripts; shell or LUA, called pre/post install
 - U-Boot; to update env variables
 - **Custom handlers**
- Streaming support: no temporary copy on the target

UPDATE IMAGE FORMAT

- `.swu` file

- CPIO format for his simplicity
- sw-description: to descript the update
- images data / update resources



EXAMPLE OF .SWU FILE

```
software =  
{  
    version = "1.0.1";  
  
    nitrogen6x = {  
        hardware- compatibility : [ "REV4" ];  
  
        images: (  
            {  
                filename = "rootfs.ext2.gz";  
                device = "/dev/update";  
                type = "raw";  
                sha256 = "1  
e0f63c1e6026acd7bba16ced9693aa862f7df04e423b668acaf9eb3fa4330a2";  
                compressed = true;  
            }  
        );  
  
        scripts : (  
            {  
                filename = "update.sh";  
                type = " shellscript ";  
                sha256 = "  
faaaa3096b01c196d20903e21ec88757834e68f09de4c2edd721ad8b83a9628e";  
            }  
        );  
    };  
}
```

Main block

Board specific

Handler for images

Handler for scripts

LOCAL UPDATE: USB EXAMPLE

- Once the drive is mounted and the update located, you can start swupdate with the -i option:

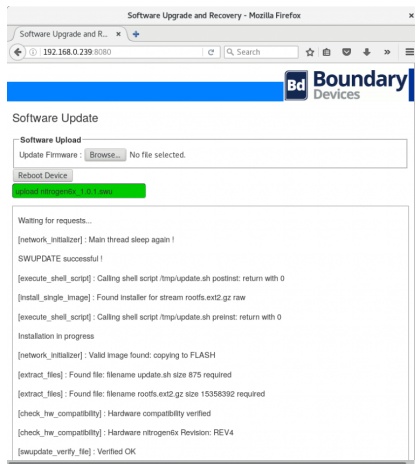
```
swupdate -i <name_of_update>
```

or

```
swupdate -i <name_of_update> -k <pubkey>
```

- In order to automate this procedure, you can create a udev/mdev rule that executes a script every time a USB drive is plugged.
 - ▶ [mdev automount - tutorial](#)
 - ▶ [hotplugging with udev - Free Electrons training](#)

OTA UPDATE: MONGOOSE



```
swupdate -k <pubkey> -w "-document_root /var/www/swupdate/"
```

OTA UPDATE: DOWNLOAD HTTP

SWUpdate also offers to download the update file from an HTTP server with the `-d` option:

```
swupdate -k <pubkey> -d <url>
```

OTA UPDATE: SURICATTA DAEMON/HAWKBIT

The screenshot shows the HawkBit web interface for Upload Management. The left sidebar contains navigation links: Deployment, Rollout, Target Filters, Distributions, Upload (selected), and System Config. The main content area is titled 'Upload Management' and includes a 'Filter by type' dropdown set to 'OS'. A table lists software modules, with 'firmware1' (version v1.0) selected. Below the table, details for 'firmware1:v1.0' are shown, including 'Vendor: OS' and 'Assignment type: Firmware (FW)'. On the right, 'Artifact Details of firmware1:v1.0' are displayed in a table with columns: File name, Size(B), Last modified date, and Action. The table contains one entry: 'nitrogenix_1.0.2.swu' (24,283,136 bytes, modified Sun Feb 19 10:34:03 PST 2017). At the bottom right, there is a dashed box with a download icon and the text 'Drop files to upload', and buttons for 'Upload File', 'Process', and 'Discard'.

File name	Size(B)	Last modified date	Action
nitrogenix_1.0.2.swu	24,283,136	Sun Feb 19 10:34:03 PST 2017	

```
swupdate -k <pubkey> -u "<hawkbit options>"
```

OTA UPDATE: SURICATTA DAEMON/HAWKBIT

The screenshot displays the HawkBit Software Provisioning web interface. The left sidebar contains navigation links: Deployment, Rollout, Target Filters, Distributions, Upload, and System Config. The main content area is titled "Deployment Management" and is divided into several panels:

- Filters:** A "Simple Filter" section with a "NO TAG" filter selected.
- Targets:** A table listing targets. Target 14 is selected, showing details: Controller Id: 14, Last poll: Sun Feb 19 10:36:09 PST 2017, Address: http://192.168.1.133, and Security token: 115yG6taGuLr10qH4.
- Distributions:** A table listing distribution sets. The "distribution" set with version "v1.0" is selected, showing details: Type: OS only and Required Migration Step: No.
- Action History For 14:** A table showing the history of actions for target 14.

At the bottom, there are buttons for "Drop here to delete" and "No actions".

Name	Status
25	Success
14	Failed

Name	Version
distribution	v1.0

Active	Distributionset	Date and time	Status	Forced	Actions
+	distributionv1.0	Sun Feb 19 10:36:09 ...	Failed	+	✖ + ✖
+	distributionv1.0	Sun Feb 19 10:32:12 ...	Failed	+	✖ + ✖

Details	Description	Attributes	Assigned
Controller Id: 14 Last poll: Sun Feb 19 10:36:09 PST 2017 Address: http://192.168.1.133 Security token: 115yG6taGuLr10qH4			

Details	Description	Modules	Tags	Log
Type: OS only Required Migration Step: No				

Docker is highly advise for tests/demos setup (even production ...)



Customization

MENUCONFIG: SELECT YOUR FEATURES

.config - Swupdate Configuration

Swrapdate Configuration

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search.
Legend: [*] built-in [] excluded <M> module < > module capable

```
||| Swupdate Settings --->
[*] Enable image downloading
[*] Enable verification of signed images
[*] Enable Suricata Daemon Mode
    Suricata --->
[*] Enable webserver
    Webserver Features --->
    Archival Features --->
    Parser Features --->
    Image Handlers --->
```

<Select> < Exit > < Help > < Save > < Load >

HANDLERS

- A way to add your own installer
- Don't forget to update handlers/lua/swupdate_handlers.lua ...

```
1 require("swupdate")
2 fpga_handler = function(image)
3     print(" Install  FPGA Software")
4     for k,l in pairs(image) do
5         print("image[" .. tostring(k) .. "] = " .. tostring(l) )
6         swupdate.notify(swupdate.RECOVERY_STATUS.RUN,0,"
7             image[" .. tostring(k) .. "] = " .. tostring(l)
8         return 0
9     end
10 swupdate.register_handler("fpga",fpga_handler)
11
```


HANDLERS

Provided by the framework:

- flash devices in raw mode (both NOR and NAND)
- UBI volumes
- raw devices, such as a SD Card partition
- U-Boot environment
- LUA scripts

But you can also create your own ...

HANDLERS: FLASH DEVICES (NOR, NAND, ...)

```
1 images:
2 {
3     "version": "2016.01",
4     "name": "bootloader",
5     "device": "mtd1",
6     "install - if - different": true,
7     "type": "flash",
8     "filename": "u-boot.sb"
9 },
10
```

HANDLERS: UBI VOLUMES

```
1 images:
2 {
3     filename = "core-image- full -cmdline. ubifs ";
4     type = "ubivol";
5     volume = "rootfs1"
6     installed - directly  = true;
7 }
8
```

HANDLERS: RAW DEVICES (SD CARD)

```
1 images:
2 {
3     filename = "rootfs.ext2.gz";
4     device = "/dev/mmcblk1p2";
5     compressed = true;
6 }
7
```

HANDLERS: U-BOOT ENVIRONMENT

```
1 images:
2 {
3     filename = "uboot-env";
4     type = "uboot";
5 },
6 ...
7 uboot:
8 {
9     name = "vram";
10    value = "4M";
11 },
12 {
13     name = "addfb";
14     value = "setenv bootargs ${bootargs} omapfb.vram=1:2M,2:2M
15             ,3:2M omapdss.def_disp=lcd"
16 }
```

HANDLERS: LUA SCRIPTS

```
1  scripts : (  
2  {  
3      filename = "erase_at_end";  
4      type = "lua";  
5  },  
6  {  
7      filename = "display_info";  
8      type = "lua";  
9  }  
10 );  
11
```

COLLECTIONS

```
1 software =
2 {
3     ...
4     stable :
5     {
6         main:
7         {
8             images: (
9             {
10                 filename = "rootfs.ext3";
11                 device = "/dev/mmcblk0p2";
12             }
13         );
14     };
15     alt :
16     {
17         images: (
18         {
19             filename = "rootfs.ext3";
20             device = "/dev/mmcblk0p1";
21         }
22         );
23     };
24 };
25 }
26 }
27
```

```
# swupdate -i /mnt/my_update.swu -e stable,alt
```



Integration

YOCTO



- meta-swupdate is provided:
<https://github.com/sbabic/meta-swupdate>
- Only 'single-copy' scheme is generated (rescue image)
- MACHINE=<your machine> bitbake swupdate-image
- Images are generated in tmp/deploy/<your machine>/
(.ext3.gz.u-boot)

YOCTO

- **bitbake bbb-swupate-image**: generate an update (.swu)
- **meta-swupdate/recipes-extended/images/bbb-swu-image.bb**: can be used as an example for your custom 'swupdate images'
- swupdate provides a **class** that can be inherit for your custom build/images

YOCTO

WARNING

Starting the swupdate initrd is platform specific ...

- Load the kernel: `load usb 0 0xDEADBEEF zImage`
- Load the device tree: `load usb 0 0xDEADFEED devicetree.dtb`
- Load the swupdate initrd: `load usb 0 0xFACEB00C swupdate-image-nitrogen6x.ext3.gz.u-boot`
- Start the update: `bootz 0xDEADBEEF 0xFACEB00C 0xDEADFEED`
- Update in progress ...

BUILDROOT



- `package/swupdate` is supported in mainline
- `BR2_PACKAGE_SWUPDATE` has to be enabled
- Default configuration provided in `package/swupdate/swupdate.config`
- Support of `menuconfig` through buildroot: `make swupdate-menuconfig`

BUILDROOT

- From the output/images/ directory, you could run the following script:

```
#!/bin/bash
```

```
CONTAINER_VER="1.0.2"
```

```
PRODUCT_NAME="sabrelite"
```

```
FILES="sw-description rootfs.ext2"
```

```
for i in $FILES;do
```

```
    echo $i;done | cpio -ov -H crc > ${PRODUCT_NAME}_${  
    CONTAINER_VER}.swu
```

BUILDROOT

```
# mount /dev/sda1 /mnt/  
# swupdate -i /mnt/my_update.swu  
# reboot
```

TARGETING A SPECIFIC HW/SW VERSION

An update can use /etc/hwrevision and /etc/swversion
(CONFIG_HW_COMPATIBILITY)

```
1 software =  
2 {  
3     version = "1.0.1";  
4     target1 = {  
5         hardware- compatibility : [ "1.0", "1.2", "1.3" ];  
6         ...  
7     };  
8     target2 = {  
9         hardware- compatibility : [ "1.1" ];  
10        ...  
11    };  
12    target3 = {  
13        ...  
14    };  
15 }  
16
```

TARGETING A SPECIFIC HW/SW VERSION

Compatible:

```
[NOTIFY] : SWUPDATE running : [check_hw_compatibility] :  
    Hardware nitrogen6x Revision: 1.2  
[NOTIFY] : SWUPDATE running : [check_hw_compatibility] :  
    Hardware compatibility verified  
[NOTIFY] : SWUPDATE running : [cpio_scan] : Found file:
```

Not compatible:

```
[NOTIFY] : SWUPDATE running : [check_hw_compatibility] :  
    Hardware nitrogen6x Revision: 1.3  
ERROR core/swupdate.c : install_from_file : 317 : SW not  
    compatible with hardware  
[NOTIFY] : SWUPDATE failed [0] ERROR core/swupdate.c :  
    install_from_file : 317 : SW not compatible with hardware
```


SIGNING AN IMAGE

Only the sw-description is **signed** but all images node **must have a hash** (sha256 for example)

Example with sha256:

- create the private key

```
openssl genrsa -out swupdate-priv.pem
```

- create the public key

```
openssl rsa -in swupdate-priv.pem -out swupdate-public.pem -  
outform PEM -pubout
```

- sign the image description file

```
openssl dgst -sha256 -sign swupdate-priv.pem sw-description  
> sw-description.sig
```

Once signed, you can ensure that nobody can temper with your update image to include malicious firmware

SIGNING AN IMAGE

```
#!/bin/bash

CONTAINER_VER="1.0.1"
PRODUCT_NAME="nitrogen6x"
FILES="sw-description sw-description.sig rootfs.ext2.gz update.sh
"

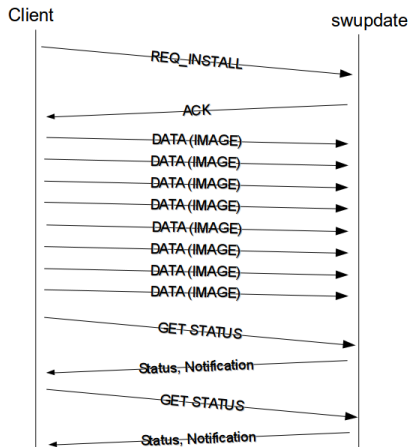
openssl dgst -sha256 -sign swupdate-priv.pem sw-description > sw-
description.sig

for i in $FILES;do
    echo $i;done | cpio -ov -H crc >  ${PRODUCT_NAME}_${
CONTAINER_VER}.swu
```

API FOR EXTERNAL PROGRAMS

- Communication via UNIX Domain Socket
- Simple interface

```
1 typedef struct {  
2     int magic;  
3     int type;  
4     msgdata data;  
5 } ipc_message;  
6
```



CONCLUSION/DEMOS

Next steps:

- Binary delta updates
- New handlers: FPGA? loading them at runtime?
- More examples and support for evaluation boards!
- More backend like Hawkbit
- Filesystem-based Persistent Update Status Storage
- ...

LINKS

- SWUpdate source code: <https://github.com/sbabic/swupdate>
- SWUpdate online documentation: <https://sbabic.github.io/swupdate>
- SWUpdate mailing list: swupdate@googlegroups.com
- Stefano Babic's ELCE presentation: [software-update-for-embedded-systems-elce2014](#)
- Boundary Devices blog: [Using SWUpdate](#)
boundarydevices.com/using-swupdate-upgrade-system
- Hawkbit docker: <https://github.com/MiloCasagrande/hawkbit-docker.git>